Authorization of XCAT Components

Introduction

Authorization is vital in a component framework. When methods are invoked on a component or connections are made it is important that only the right people can do it. The three main functions in the component framework are

1. Instantiate a component
2. Connecting component
3. Invoke a Method

Issues in Component Interaction

There are some issues to be considered to set up an authorization policy. We may not have answers to all of them at the given point of time but they are essential to implement a policy.

1. The granularity that the authorization needs to be implemented at. The two that are being considered are port level and method level. The port level granularity tends to put the responsibility on the user to place methods with similar security policies in the same port. At the method level the overhead of the authorization check may need to be considered.

2. Where is the authorization policy likely to be stored? In a centralized approach an LDAP or a similar registry service may be considered. Though maintenance of security policies may be easier in this case there are a couple of other issues. A centralized approach tends to be slower and also the need for authenticating and authorizing access to this database is to be considered. In a distributed solution each component itself has this information. It can be a simple access control list or a more complicated policy.

3. Representation of the authorization policy will need to be considered. Will the policy be a simple access control list, a role based system, etc.

4. There could be time -sensitive material in the components. Now Component A may like to allow Component B to access its ports/methods only till Tuesday September 15th. When Component B connects to A on September 13th and remains connected for an extended period of time. So this connection will need to be revoked at the appropriate time. This may be simple if we could poll at the end of the day and revoke all such connections that fail the authorization check. But now suppose Component B will like Component A to be connected only for 8 hours from start of connection. Now this same polling will need to be done more frequently to revoke this connection. A countdown timer may also be an option.

Instantiation of Components

When you instantiate a component, GRAM already takes care of the authentication and authorization. If you don't have access to a machine then you cant launch a component on that so authorization has been taken care of. It uses your globus certificate and checks it up with the grid map file. The underlying assumption here is that anyone who has access to a particular machine can instantiate components on it.

Policy Initialization

The authorization policy will need to be initialized at some point of time. When a component is created it should be able to specify if it wants to implement an authorized policy or would choose to allow unrestricted access i.e. not implement any security policy. If it chooses to be secure then it also needs to initialize its policy of who is allowed access to what. It seems appropriate to do this initialization at creation time. Also by default a component that instantiates other components should have unrestricted access to it since it is the owner of the component. Of course it should be easy to even override this policy. Another scenario that may need to be considered is that we may want to specify the same security policy for an experiment environment or include details on where this policy may be allowed to vary. Thus there is a need to be able to represent a global policy too.

Connection of Components

There are a couple of scenarios that may help decide the kind of policy that may be required.

1. Component A connects to Component B

a) Component A or B was instantiated by the other.

By the default policy since they are in the same hierarchy tree (two components are considered to be in the same hierarchy tree when they have been either instantiated by the other or by a common third component.) they should be allowed to connect to each other.

b) They are two hierarchically independent components

Here we need a clear policy on whether the components are allowed to connect to each other or not.

2. Component A connects Component B to Component C

a) Component A, B and C belong to the same hierarchy tree

By the default policy since they are in the same hierarchy tree they should be allowed to connect to each other otherwise it will be covered by the general connection policy.

b) Component B or C was instantiated by A

Now when a third component is trying to instantiate the connection process it needs to find out whether both the parties involved are ready to participate in the connection. Now if A instantiated one of the components (say B) it may be okay to assume that B is willing to participate in the assumption.

c) Component B and C belong to the same hierarchy tree whereas A does not.

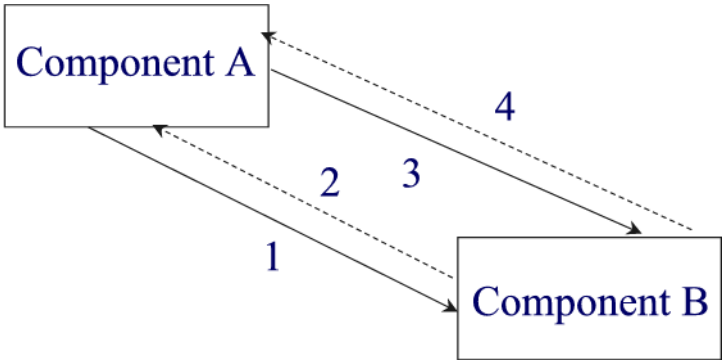
Now here B and C will be ready to connect to each other since they are part of the same hierarchy tree but A may not know that B and C are of the same hierarchy tree.

d) They are all hierarchically independent components

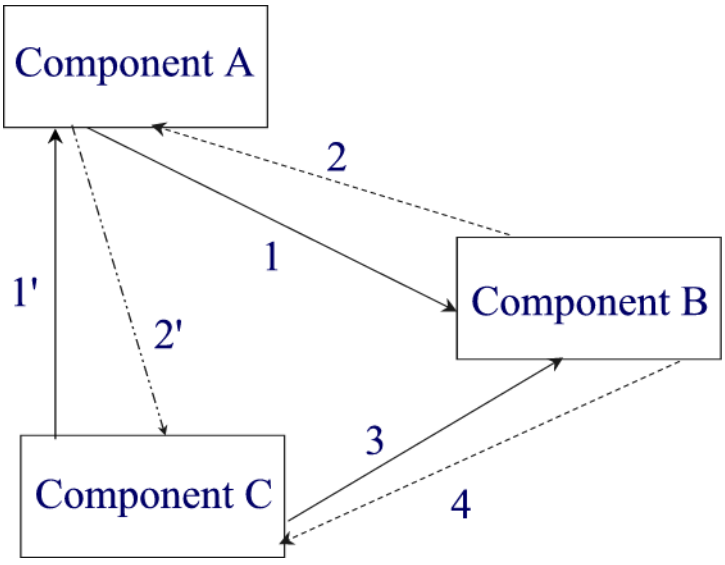
Now at connection time the component that instantiates the connection needs to find out that the Component that is providing the service is ready to accept connections from another component. It is safe to assume that the component that wants to use the service will be ready to

participate in the connection. So the component that instantiates the connecting makes an authenticated (this is done at the lower SoapRMI level) connection to the Provider Component to make sure it is ready for the connection. The identity of the component that wishes to connect is passed during this call. The Provider Component in turn will look into its authorization policy and either grant the connection or reject it. If the User Component is not allowed any method invocation on the
Component it will outright reject it. Otherwise it will send a kind of ticket back that can be used when methods are invoked. The component that instantiates the connection will need to pass this onto the user component if different from itself.

Examples:



In this Scenario Component A wishes to connect to Component B. The sequence of steps can be represented as
1. Component A connects to B to find authorization policy during connect.
2. Component B sends back a ticket to A that A cannot use when it wants to invoke any methods on B
3. Component A invokes a method on Component B. It also sends the ticket it got from B.
4. B now checks if A has a valid ticket for the method invoked and returns accordingly.



In this Scenario Component A wishes to connect to Component B to Component C. The additional steps that may be required in this scenario that are different from earlier scenario are

1’ Component C requests Component A to connect it to A. (this may not be always true since A may decide to connect C to B on its own accord.)
2’ Component A sends the ticket to C so that it can invoke methods on B when required.

At Method Invocation Time
When a user component needs to invoke a method it passes the ticket that it got earlier when connection was established. The Provider Component checks to see if the ticket is valid and whether this particular method is allowed. If it is allowed then the invocation can take place.

Interaction of Un-secure and Secure Components
When a component that implements security policy tries to connect to a un-secure Provider Component it will be allowed to access all its methods without any restriction since no authorization policy is equivalent to a free - for - all policy. If an un-secure Component will like to interact with a component that has an authorization policy it will need to be able implement a partial security policy i.e. to authenticate itself, accept the ticket that the Provider Component sends back and send that ticket back when it wants to invoke a method.

Form of identity used
The semantics of determining the identity of the component needs to be chalked out. One possible solution is to recognize a Component by the user who instantiates it. But then the authorization policy will be based on the user identity rather than the component function. Also the format of the tickets passed around need to be amongst other things be such that
1. Each ticket issued on each connection should be different from all the rest in the environment
2. The tickets are not tampered before they are passed back.
3. It should not be possible to impersonate the issuer

***Note**: Components as mentioned in this document refer to scientific and engineering components that keep up the CCA forum specification (http://www.acl.lanl.gov/cca/)*